特别提示：本代码为所有顶点的数据类型由整型改为双精度后代码，这样可避免因类型转换引起精度损失。其中被替代的代码仍保留，但已被注释掉。
功能：B样条曲线的一般分割，插入区间左端点a和右端点b,求出定义在该区间上那段子B样条曲线的控制顶点与节点矢量。
输入参数：(m_xAVertex,m_yAVertex)-控制顶点，m_aNode-节点矢量，m_nTimes-次数，都是受保护成员。 a,b-区间左端点和右端点。
输出参数：(m_xnewVertex,m_ynewVertex)-子B样条曲线的控制顶点，m_newNode-子B样条曲线的节点矢量。
调用函数：GetKnotsRepeats-给出节点矢量的两个数组表示。

```
void InsertAB(double a,double b)
{
        int aMultiple=0,bMultiple=0,la,lb;
//      CArray<CPoint,CPoint> LeftVertex;
        CArray<double,double> xLeftVertex,yLeftVertex;
//      CArray<CPoint,CPoint> Temp;
        CArray<double,double> xTemp,yTemp;
        CArray<double,double> LefNode;
        int k=m_nTimes;
        xTemp.SetSize(k+1);  yTemp.SetSize(k+1);
        int ia,ib;
        //节点矢量的首末节点取值对曲线在定义域首末端点所取矢量值没有影响,
        //下面两语句处理后使得重复度aMultiple与bMultiple计算对定义域端节点与内节点统一起来,
        //且使得ia与ib的确定，不论ia=0或ia>0, 也不论b<1或b=1, 都统一起来。若b=1,则ib=Node.GetSize()-2.
        m_aNode[0]=0.-1.e-15;
        m_aNode[m_aNode.GetSize()-1]=1.+1.e-15;

        for(int i=0;i<m_aNode.GetSize();i++)
        {
                if(a==m_aNode[i])
                {
                        aMultiple=aMultiple+1;       //插入a处已有重复度
                }
                if(a!=m_aNode[i]&&aMultiple!=0) break;
        }
        for(i=0;i<m_aNode.GetSize();i++)
        {
                if(a<m_aNode[i]) {ia=i-1;break;} //ia-a所在节点区间左端点下标
        }
        for(i=0;i<m_aNode.GetSize();i++)
        {
                if(b==m_aNode[i])
                {
                        bMultiple=bMultiple+1;       //插入b处已有重复度
                }
                if(b!=m_aNode[i]&&bMultiple!=0) break;
        }
        for(i=0;i<m_aNode.GetSize();i++)
        {
                if(b<m_aNode[i]) {ib=i-1; break;} //ib-b所在节点区间左端点下标
        }
        la=k-aMultiple;                       //需插入a的次数
        lb=k-bMultiple;                       //需插入b的次数

        if(aMultiple>=k) la=0;
        if(bMultiple>=k) lb=0;
        if(b==1&&bMultiple>k) ib--;
        xLeftVertex.SetSize(ib+lb-k+1);  yLeftVertex.SetSize(ib+lb-k+1);
        LefNode.SetSize(ib+lb+2);
//      for(int j=0;j<=ib+lb-k;j++) LeftVertex[j]=m_aVertex[j];
        for(int j=0;j<=ib+lb-k;j++) {xLeftVertex[j]=m_xAVertex[j];  yLeftVertex[j]=m_yAVertex[j];}
        for(j=0;j<=ib+lb+1;j++) LefNode[j]=m_aNode[j];
        if(lb>0&&b<=1.)
        {
                for(int j=0;j<=k-bMultiple;j++)
//                      Temp[j]=LeftVertex[ib-k+j];
                {xTemp[j]=xLeftVertex[ib-k+j];  yTemp[j]=yLeftVertex[ib-k+j];}
                for(int s=1;s<=lb;s++)
                {
                        for(int j=ib-k;j<=ib-bMultiple-s;j++)
                        {
                                double alpha;
                                if((LefNode[j+k+1]-LefNode[j+s])==0.) alpha=0.;
                                else alpha=(b-LefNode[j+s])/(LefNode[j+k+1]-LefNode[j+s]);
//                              Temp[j-ib+k].x=int((1-alpha)*Temp[j-ib+k].x+alpha*Temp[j-ib+k+1].x);
//                              Temp[j-ib+k].y=int((1-alpha)*Temp[j-ib+k].y+alpha*Temp[j-ib+k+1].y);
                                xTemp[j-ib+k]=(1-alpha)*xTemp[j-ib+k]+alpha*xTemp[j-ib+k+1];
                                yTemp[j-ib+k]=(1-alpha)*yTemp[j-ib+k]+alpha*yTemp[j-ib+k+1];
                        }
//                      LeftVertex[ib-k+s]=Temp[0];
                        xLeftVertex[ib-k+s]=xTemp[0];  yLeftVertex[ib-k+s]=yTemp[0];
                }
                for(j=ib+lb-k+1;j<=ib+lb+1;j++)
                        LefNode[j]=b;
        }
        if(la>0)
        {
                int j;
                for(j=0;j<=k-aMultiple;j++)
                {
//                      Temp[j]=LeftVertex[ia-k+j];
                        xTemp[j]=xLeftVertex[ia-k+j];
                        yTemp[j]=yLeftVertex[ia-k+j];
                }
                for(int s=1;s<=la;s++)
                {
                        for(j=ia-k;j<=ia-aMultiple-s;j++)
                        {
                                double alpha;
                                if((LefNode[j+k+1]-LefNode[j+s])==0.) alpha=0.;
                                else alpha=(a-LefNode[j+s])/(LefNode[j+k+1]-LefNode[j+s]);
//                              Temp[j-ia+k].x=int((1-alpha)*Temp[j-ia+k].x+alpha*Temp[j-ia+k+1].x);
//                              Temp[j-ia+k].y=int((1-alpha)*Temp[j-ia+k].y+alpha*Temp[j-ia+k+1].y);
                                xTemp[j-ia+k]=(1-alpha)*xTemp[j-ia+k]+alpha*xTemp[j-ia+k+1];
                                yTemp[j-ia+k]=(1-alpha)*yTemp[j-ia+k]+alpha*yTemp[j-ia+k+1];
                        }
                }
                for(j=ia-k;j<=ia-k+la;j++)
                {
//                      LeftVertex[j]=Temp[j-ia+k];
                        xLeftVertex[j]=xTemp[j-ia+k];  yLeftVertex[j]=yTemp[j-ia+k];
                }
        }
//      m_newVertex.SetSize(ib-ia+lb+1);
        m_xnewVertex.SetSize(ib-ia+lb+1);  m_ynewVertex.SetSize(ib-ia+lb+1);
        for(j=ia-k;j<=ib+lb-k;j++)
        {
//              m_newVertex[j-ia+k]=LeftVertex[j];
                m_xnewVertex[j-ia+k]=xLeftVertex[j];  m_ynewVertex[j-ia+k]=yLeftVertex[j];
        }
        if(ia==ib)
        {
                m_newNode.SetSize(2*k+2);
                for(j=0;j<=k;j++) m_newNode[j]=0.;
                for(j=k+1;j<=2*k+1;j++) m_newNode[j]=1.;
        }
        if(ib>=ia+bMultiple)
        {
                if(bMultiple==0)
                {
                        m_newNode.SetSize(ib-ia+2*k+2);
                        for(j=0;j<=k;j++) m_newNode[j]=0.;
//                      for(j=k+1;j<=k+ib-ia;j++) m_newNode[j]=(m_aNode[ia+j]-a)/(b-a);
                        for(j=k+1;j<=k+ib-ia;j++) m_newNode[j]=(m_aNode[ia+j-k]-a)/(b-a);
                        for(j=ib-ia+k+1;j<=ib-ia+2*k+1;j++) m_newNode[j]=1.;
                }
                if(bMultiple>0)
                {
//                      m_newNode.SetSize(ib-ia+k+2);
                        m_newNode.SetSize(ib-ia+2*k+1);
                        for(j=0;j<=k;j++) m_newNode[j]=0.;
//                      for(j=k+1;j<=ib-ia;j++) m_newNode[k+j]=(m_aNode[ia+j]-a)/(b-a);
                        for(j=k+1;j<=ib-ia+k-1;j++) m_newNode[j]=(m_aNode[ia+j-k]-a)/(b-a);
//                      for(j=ib-ia+1;j<=ib-ia+k+1;j++) m_newNode[j]=1.;
                        for(j=ib-ia+k;j<=ib-ia+2*k;j++) m_newNode[j]=1.;
                }
        }
        int n=m_newNode.GetSize()-k-2;
        CArray<double,double> TemNode;
        TemNode.SetSize(m_newNode.GetSize());
        for(j=0;j<m_newNode.GetSize();j++) TemNode[j]=m_newNode[j];
        GetKnotsRepeats(k,n,TemNode);
}
```